Amendments to the Claims

1. (currently amended) A network proxy server[[,]] comprising:

a network connection configured to receive ~~able to intercept~~ content-object requests generated by a plurality of clients, said content-object requests requesting a content-object from a server~~of clients from a server, and able to respond instead of said server to such; and~~

a plurality of moving-window buffers coupled with said network connection, said plurality of moving-window buffers being configured to service said content-object requests; and

first and second content buffers coupled with said network connection, said first content buffer being configured to duplicate a first portion of a content passing from said server to said plurality of clients, cache said first portion, and provide said first portion to a subsequent client in response to a request for said first portion, and said second content buffer being configured to duplicate a second portion of said content and cache said second portion, and wherein said first and second content buffers are further configured to simultaneously provide said first and second portions of said content to said subsequent client in response to a request for said first and second portions.

~~a plurality of content buffers for duplicating web content passing through from said server to any client, and for caching such web content to any subsequent clients;~~

~~wherein, multiple, moving-window buffers are included in the plurality of content buffers to service content requests of a server by various independent clients; and~~

~~wherein, requests for content-object from single clients can be serviced simultaneously from parts distributed across more than one such content buffer.~~

2. (currently amended) A method ~~system~~ of delivering objects from servers to clients, said method comprising:

receiving a first request for [[an]]a content object from a first client;

allocating a first running buffer;

retrieving the content object as a datastream having a start point and inserting the datastream into the first running buffer while delivering the ~~same~~ datastream to the first client;

when the first running buffer is filled, deleting data from the start point of the datastream while continuing to insert retrieved data into the first running buffer[[,]] so that the first running buffer contains a moving window of the retrieved data;

receiving a second request for the content object from a second client;

if the second request is received while the start point of the datastream is still in the first running buffer, serving the content object directly from the first running buffer; and

if the second request is received after the start point has been deleted from the first running buffer, retrieving a ~~the~~ portion of the content object that has

been deleted from the first running buffer, commencing from the start point, and delivering the ~~same as a~~ datastream while simultaneously delivering a different part of the content object from the first running buffer.

3. (currently amended) The method ~~system~~ of claim 2, further comprising[[,]]:

~~if the second request is received after the start point of the datastream has been deleted from the first buffer:~~

allocating a second running buffer when the second request is received after the start point of the datastream has been deleted from the first running buffer and inserting a ~~the datastream representing the~~ portion of the content object not in the first running buffer into the second running buffer while delivering the ~~same~~ datastream.

4. (currently amended) The method ~~system~~ of claim 3, further comprising:

receiving ~~for~~ a third request for the content object ~~received~~ after the second running buffer has been allocated;

checking whether the start point is cached in an existing running buffer;

if the start point is cached in an existing running buffer, serving the content object as a datastream from each of the running buffers simultaneously;

if the start point is not cached in an existing running buffer, allocating a third running buffer; and

retrieving a ~~the~~ portion of the content object not in an existing running buffer as a datastream and inserting the datastream into the third running buffer while delivering the ~~same~~ datastream and simultaneously delivering a different part of the content object from other existing running buffers.

5. (currently amended) The method system of claim 2, further comprising:

determining ~~wherein the first buffer or another buffer has~~ a size of the first buffer ~~that is determined~~ as a proportion of an advertised length of the content object.

6. (currently amended) The method ~~system~~ of claim 2, further comprising:

modifying a ~~the~~ size of the first buffer or another buffer in response to an analysis of a frequency of requests for the content object[[,]] in order to optimize an allocation of a memory.

7. (currently amended) The method ~~system~~ of claim 2, further comprising[[,]]:

prior to allocating the first buffer ~~or another buffer~~, applying a replacement algorithm to reclaim buffers from less frequently requested objects.

8. (currently amended) The method ~~system~~ of claim 2, wherein the content object has a time length [[L]]and each of the first and second running buffers ~~each buffer~~ has a start time, [[S_i,]]an end time [[E_i]]and a running distance [[D_i]], said method further comprising:

allocating a second running buffer, wherein a ~~the~~ running distance ~~D_i for each buffer after the first buffer equals: $D_i=S_i-S_{i-1}$,~~ for the second running buffer equals a distance in time between a start time of the second running buffer and a start time of the first running buffer, and wherein an ~~the~~ end time of the second running buffer equals a lesser of (1) a distance in time between the start time of the first running buffer and the running distance of the second running buffer and (2) the start time of the second running filter plus the time length for the content object~~E_i for each buffer after the first buffer is, $E_i=\min(S_{latest}+D_i, S_i+L)$, where, S_{latest} is the start time of the most recent buffer allocated~~.

9. (currently amended) A computer-implemented method of shared running-buffer-based caching, said computer-implemented method comprising:

~~Computer data storage media having stored thereon software performing the following functions:~~

receiving a first request for [[an]]a content object;

allocating a first running buffer;

retrieving the content object as a datastream having a start point and inserting the datastream into the first running buffer while delivering the ~~same~~ datastream;

when the first running buffer is filled, deleting data from the start point of the datastream while continuing to insert retrieved data into the first running buffer, so that the first running buffer contains a moving window of the retrieved data;

receiving a second request for the content object;

if the second request is received while the start point of the datastream is in the first running buffer, serving the content object directly from the first running buffer;

if the second request is received after the start point has been deleted from the first running buffer:

retrieving a ~~the~~ portion of the content object that has been deleted from the first running buffer, commencing from the start point, and delivering the ~~same as a~~ datastream while simultaneously delivering a different part of the content object as a datastream from the first running buffer.

10. (currently amended) The computer-implemented method of claim 9, further comprising:

~~The computer-data-storage-media of claim 9, wherein the software performs the following further functions:~~

if the second request is received after the start point of the datastream has been deleted from the first running buffer, allocating a second running buffer and inserting the datastream representing a ~~the~~ portion of the content object not in

the first running buffer into the second running buffer while delivering the ~~same~~ datastream.

11. (currently amended) The computer-implemented method of claim 9, further comprising:

~~The computer data storage media of claim 9, wherein the software performs the following further functions:~~

receiving a third request for the content object after the second running buffer has been allocated;

checking whether the start point is cached in an existing running buffer;

if the start point is cached in an existing running buffer, serving the content object as a datastream from each of the running buffers simultaneously;

if the start point is not cached in an existing running buffer:

allocating a third running buffer; and

retrieving a ~~the~~ portion of the content object not in an existing running buffer as a datastream and inserting the datastream into the third running buffer while delivering the ~~same~~ datastream and simultaneously delivering a different part of the content object as a datastream from other existing running buffers.

12. (currently amended) The computer-implemented method of claim 9, further comprising:

~~The computer data storage media of claim 9, wherein the software performs the following further functions:~~

determining an ~~the~~ advertised length of the content object; and

setting a ~~the~~ size of the first buffer ~~or another buffer~~ as a proportion of the ~~an~~ advertised length of the content object.

13. (currently amended) The computer-implemented method of claim 9, further comprising:

~~The computer data storage media of claim 9, wherein:~~

analyzing a frequency of requests for the content object; and

modifying a ~~the~~ size of the first buffer ~~or another buffer~~ in response to the analyzing ~~analysis~~ of the frequency of requests for the content object in order to optimize an allocation of a memory.

14. (currently amended) The computer-implemented method of claim 9, further comprising:

~~The computer data storage media of claim 9, wherein:~~

prior to allocating a buffer, ~~the first buffer or another buffer~~ checking if a sufficient memory for allocating the buffer is available; and

if the sufficient memory is not available~~there is not enough memory available to allocate a buffer~~, applying a replacement algorithm to reclaim buffers from less frequently requested objects.

15. (currently amended) The computer-implemented method of claim 9, further comprising:

~~The computer-data-storage-media of claim 9, wherein:~~

determining a time length [[L]]for the content object;

allocating a second running buffer; and

setting a start time, [[S_i,]]an end time [[E_i]]and a running distance [[D_i]]for each of the first and second running buffers~~buffer;~~, wherein a running distance for the second running buffer equals a distance in time between a start time of the second running buffer and a start time of the first running buffer, and wherein an end time of the second running buffer equals a lesser of (1) a distance in time between the start time of the first running buffer and the running distance of the second running buffer and (2) the start time of the second running filter plus the time length for the content object.

~~computing the running distance D_i for each buffer after the first buffer as, $D_i=S_i-S_{i-1}$;~~

~~computing the end time E_i for each buffer after the first buffer as, $E_i=\min(S_{latest}+D_i, S_i+L)$, where, S_{latest} is the start time of the most recent buffer allocated.~~